EECS 150 Spring 2004

Checkpoint1 Part2

3/5/2004

Greg Gibeling

Today

- Finer Points of SDRAM
- Common Mistakes
  - A little verilog style
- Announcements
- Synplify Warnings
- Design Reviews

Finer Points of SDRAM (1)

- Errata
  - Your testbench needs two SDRAM instances (there are two SDRAM chips)
  - The write FIFO needs 4 words before the first write
  - Programmable Delay/Time Select is in 10msec units
  - There was a bug in PNGen, sorry…

Finer Points of SDRAM (2)

- Write and Read are independent
  - SDRAM controller is also a bus arbiter
  - Decides whether to do a read or a write
  - Schedules access to the data bus
    - Can't read and write on the same cycle
    - But read and write requests might come in at the same time
  - This will be necessary later in the project

Dividing the Checkpoint

- SDRAM Control
  - Handles sending SDRAM commands
  - Coordinates reads and writes
  - Dictates when data is required/ready
- SDRAM Top
  - Controls PNGens
  - Controls FIFOs
  - Basically everything specific to the checkpoint

SDRAM Clocking (1)

- There is MAJOR clock skew
  - RAM_CLK is delayed on its way to SDRAM
- How do we cope? Add more skew…
  - assign RAM_CLK = ~Clock;
  - And a negedge register between SDRAM and FIFO2 (not between FIFO1 and SDRAM)

SDRAM Clocking (2)

```
always @ (negedge Clock)
begin
        register <= RAM_D_IN;
end
```

- Why no reset? It's just added for delay…
- 1cycle (neg-neg) from SDRAM to register
- 1/2cycle (neg-pos) from register to FIFO2

Common Mistakes (1)

- Verilog Style
  - Remember Lab Lecture #4?
  - WE CANNOT DEBUG POOR VERILOG
  - (Neither can you)
- Please remember:
  - Always @ Posedge <= NonBlocking
  - Always @ (*) = Blocking
  - You cannot substitute one for the other

Common Mistakes (2)

- If you don't know what circuit you want how should synplify know?
- Verilog meant for simulation
  - Not everything synthesizes
- Simulation vs Circuit

For Loops	Counter
Initial blocks	Reset
# for delays	Properly sized counter
Integers	Use wires or regs

Announcements (1)

- Deadlines
  - Checkpoint1
    - Part1: Should be done by now
    - Part2: Must be done this coming week
  - Don't fall behind
  - 2 weeks seems like forever, but you'll run out of time later

Announcements (2)

- We will begin design reviews this week
  - 10-15min weekly appointment with a TA (pick ONE)
  - Bring diagrams!
  - This is just to make sure you're on the right track
  - Diagrams will be returned within 24hrs with a 0,1, or 2 grade
- Homework will be getting easier

Synplify Warnings (1)

- PART OF YOUR PROJECT GRADE
- Knowing these will make your life easier
- Incomplete Sensitivity
  - ModelSim will use the sensitivity list
  - Synplify pretty much ignores it

Synplify Warnings (2)

```
input          Clock;
reg    [31:0] Count;
// Counter                          OK!
always @ (posedge Clock)
        Count <= Count + 1;

input  [15:0] A, B;
output[31:0] Sum;
output         COut;                Incomplete
// Adder                           Sensitivity
always @ (A or B)
        {Sum, COut} = A + B;
```

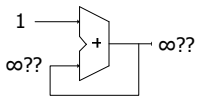
Synplify Warnings (3)

- Latch Generated

```
input    [1:0]    select;
input             A, B, C;
output            Out;
reg               Out;
// Mux
always @ (select or A or B or C) begin
        case (select)
                2'b00: Out = A;
                2'b01: Out = B;
                2'b10: Out = C;
                default: Out = 1'bx;
        endcase
end
```

Synplify Warnings (4)

- Combinational Loop



- Must remove the loop or add a register
- Synplify wont show you the loop

Synplify Warnings (5)

- FPGA_TOP(2) always has warnings
  - Undriven Input
  - Unconnected Output
  - These are truly unneeded pins
    - Things like the video chips…
- In your modules these are a problem
  - Synplify will optimize your design
  - Unconnected modules removed

Synplify Warnings (6)

- Synplify warnings will result in lost points
- Getting rid of warnings will ease debugging
- Warnings are the only syntax check
  - Verilog is a forgiving language
  - This isn't a good thing

Design Reviews

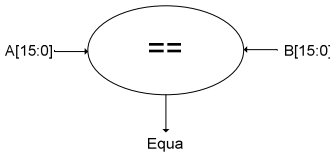
- Show design to TA
- Explaining will help you refine and understand the design
- High Level Block Diagrams
- State Transition Diagrams
  - We have trouble reading your verilog
  - Block Diagrams make the world go round

Basic Digital Building Blocks

- Combinational (Unclocked, Logic)
  - AND/NAND/OR/NOR/NOT/XOR/XNOR
  - **Comparator**
  - Adder
  - Next State/Output Functions
  - NO MULTIPLY OR DIVIDE (Shift Instead)
- Register (Sequential, Clocked)
  - **Shift Register**
  - **Counter**
  - **FIFOs**
  - Memory
  - State Register

Combinational Blocks

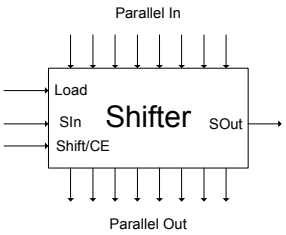
- Comparator
  - Equality is cheap
  - Greater-than/Less-than are EXPENSIVE




```
assign equal = (A == B);
```

Sequential Blocks (1)

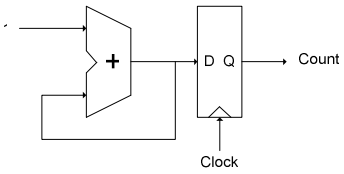
- Shift Registers
  - Parallel to Serial
  - Serial to Parallel



Sequential Blocks (2)

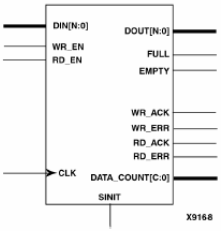
- Counter

```
input          Clock;
reg    [31:0] Count;
// Counter
always @ (posedge Clock)
        Count <= Count + 1;
```



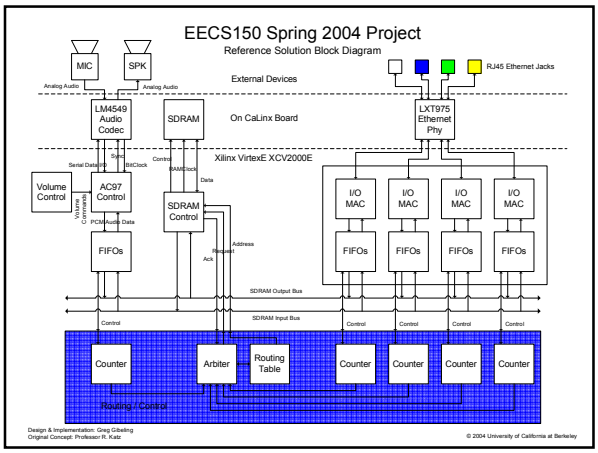
Sequential Blocks (3)

- FIFOs
  - Buffer to match two data rates
  - Great for datapath clock domain crossings



FSM Blocks / Controllers

- Clearly specify the control signals
- Mark which FSM outputs them
- Leave control connections off block diagram
- ASMD
- Bubble and Arc


EECS150 Spring 2004 Project
Reference Solution Block Diagram
MIC
SPK
Analog Audio
Analog Audio
External Devices
RJ45 Ethernet Jacks
LM4549 Audio Codec
SDRAM
On CaLinx Board
LXT975 Ethernet Phy
Xilinx VirtexE XCV2000E
Serial Data In/Out
Sync
BitClock
Control
RAMClock
Data
Volume Control
Volume Commands
AC97 Control
PCM Audio Data
SDRAM Control
Address
Request
Ack
FIFOs
I/O MAC
FIFOs
SDRAM Output Bus
SDRAM Input Bus
Control
Counter
Arbiter
Routing Table
Routing / Control
Design & Implementation: Greg Gibeling
Original Concept: Professor R. Katz
© 2004 University of California at Berkeley